

# Introduction

## Before Purchase

If you are trying to decide whether to buy this book please read far enough to reach a conclusion. I will do my very best to help you reach the right conclusion for you because delighted though I would be to have vast sales figures I do not want you to waste your time and money buying something you later regret.

This is a unique book on the subject of computer programming because it has been written for ordinary people and it attempts, I believe successfully, to make programming accessible to anyone with a computer (at this stage, one running some version of Microsoft Windows), some curiosity about what programming is and the willingness to spend some time satisfying that curiosity by learning to program.

This book is a collaboration between me as a technically knowledgeable and experienced teacher and Roberta, whose qualifications were exactly those that a reader will need. Roberta's contribution is small in textual content and vast in helping me to write a book that can be used by someone whose computing skills are just enough to load a program, use a word processor, use email and surf the Internet. When she started as the student half of the authorial team, despite having used a computer for a decade she still had not grasped the concepts of directory structures and the like. Her study of mathematics ended at 16 and her mathematical skills more or less stop with simple arithmetic and those skills needed to keep a set of company accounts.

She had two positive qualifications; she wanted to discover what programming was about and she was willing to trust me to show her. Both those are important. If you use this book you will need both those qualifications. You need to be willing to put in time and effort to discover the rudiments of programming and you need to trust us, Roberta and me, to help you achieve that ambition. However given those qualifications we promise you that you can learn to program and that long before you finish this book you will have written programs for yourself. As long as you have some imagination some of those programs will be uniquely yours. Roberta had written her first entirely original program before she had finished Chapter 6 and by the time she had finished her studies she had written several programs for her grandchildren as well as at least one following her own interests.

I tell you these things because I am certain that anyone who wants to can learn simple programming. I also believe that many people will find programming rewarding in many ways. One of those is the tremendous sense of achievement that any programmer gets whenever a program finally works and does what it is designed to do.

If you browse through the pages of this book you may wonder if you could ever cope with the weird things written in this font. There is no need to worry, you will soon find that all that text is just a way to express intentions in a way that a computer can use, and that it isn't at all weird. It isn't English though there is a scattering of English words in it. It is a computer language called C++ (pronounced cee plus plus). Friends, relatives and colleagues who know something about programming may

give you dire warnings on hearing that this book uses C++. Believe me, they are well intentioned but mistaken.

## Why C++?

Let me ask you a different question, “Why English?” Well you know the answer to that; it is a language you speak. Think a little further, what is the most widely spoken human language? Chinese is the mother tongue for more people than any other language, so why am I not writing in Chinese? On the other hand languages like Spanish and Swahili are far easier to learn than English so why am I not writing in one of those?

As you know, English is not only the mother tongue of a few hundred million people but it is also the second language for immensely more people. If you were an alien visitor to Earth I doubt that you would think twice about which human language you should start with. For all its complexity English is overwhelmingly the first choice language for those who want to move outside their own community.

C++ is very like that in the computing community. It is a rich and complex language with dark corners and traps for the unwary. But it is also the most widely used general-purpose computer programming language. Few people, if any, ever master the whole of English and few people, if any, master the whole of C++. But we do not need mastery of the whole of English nor do we need mastery of the whole of C++. This is not a book about C++ and when you finish it you will not be a C++ programmer. What you will be is a programmer who can use C++ to express solutions to problems and to write programs that meet real needs.

Why C++? Exactly because C++ does not get in the way of my showing you how to program. Other programming languages may be simpler but too often I would find myself frustrated because they would prevent me from showing you simple answers to programming problems. I have been able to pick and choose from the richness of C++ to empower my readers with powerful tools that match powerful ideas.

C++ has one small failing in that the basic language lacks tools for graphical work. That was easily fixed because I could write those tools in C++ and make them available to you. I needed some specialist help with those tools because of the quiriness of computers: they have different graphical facilities, numbers of colors on the screen, etc. Using C++ allowed me to specify what I needed and have a colleague (Garry Lancaster) turn those specifications into tools that will work on all MS Windows machines. Eventually (maybe even before you see this book) I will find others who can turn that C++ into identical tools for other machines but until I do, we have to put up with an artificial limitation in that programs you write using my tools will only work on machines running some version of MS Windows.

The last element I needed was some simple tools for you to use to write programs and manage the various technical details of turning what you write into something the computer can use. Those tools were provided by another writer, Al Stevens, who gave me permission to distribute Quincy, which is his tool set for newcomers to programming.

The work of people like Garry Lancaster and Al Stevens demonstrates the very best of the computing community, good work freely shared. The consequence is that you have in your hands everything you need (other than a computer and your time and energy) to learn to program.

The choice is entirely yours, if you want to learn to program and by doing so learn a bit about how other people’s computer programs work, you can. Roberta and I have spent nine solid months writing this book for you (that does not mean it will take you nine months to read it – she had the added burden of persuading me to improve the text so that others would find it easier). For the first time you have a real choice about learning to program. We have done our, bit the rest is up to you.

## Getting the Best from This Book

Now you have decided to buy this book let me give you some advice on how to get the best use out of it.

Ideally you should not study alone. Note that I wrote “ideally”, in practice you may find that you have no choice other than to study by yourself. However, avoid that option if you can. With that in mind the following is offered as, I hope, helpful advice rather than as some requirement for studying this book.

Two things will help you, a partner and a mentor. The partner should be someone of similar ability and someone with whom you are happy to learn, someone with whom you can share your mistakes as well as your successes. The process of learning includes making mistakes. Mistakes are nothing to be ashamed of; they are the way we learn. We should feel comfortable with sharing our mistakes with a partner in learning. Sometimes we may laugh at our idiocy, and sometimes we may be impressed at the insight of our study partner. What we should never do is laugh at someone else. The mistake may be cause for laughter but the person making it deserves respect for letting you learn from their mistakes.

If you do not have someone you know who wants to learn with you, it is worth seeing if you can contact someone via the Internet. Please check the book's website where you will find links to potential sources of study partners. To get to the book's website use the file on the CD (copied to your hard-drive when you install it) called "Link to Website.html". For success, you should be comfortable with your study partner and broaden the base of the relationship so that you do not just communicate about technical programming issues. Regular human contact even if only via email is more important to learning than most people realize. The other person needs to be considered as just that, a person.

I am also providing you with a virtual partner, my assistant author. She is the person who was first to read every word that is here as well as many that were omitted because she found them unhelpful. Her comments and experiences with each chapter are included. Sometimes her questions and my answers have been included as well. At the end of this introduction she will add a short section introducing herself and from then onwards she will be that vital second set of eyes that every technical writer should have. If this book is easy to read, you have her to thank. If you still find some of it hard going remember that she has been there ahead of you making the path a little easier.

The second person to help you, a mentor, should be an expert who can correct you when you stray off course, encourage you to persevere and compliment you on your successes. A good mentor is an invaluable resource; a bad one is a disaster.

You will identify the bad ones pretty quickly because they will want to tell you all kinds of things that are not in the pages of this book. The poor ones will want you to start from where they are or have you learn the way they did. A mentor who does not give helpful correction and reassurance as to your progress is a waste of time. One quality of a good mentor is that they are willing to learn from your work as well as to guide you. In other words they are true experts, always hungry for new ideas, new viewpoints and new insights. I loved teaching not only for what I could teach my students but also for what they could teach me.

If you cannot find a suitable mentor, try the book's website again. You can also try doing without (certainly better than having a poor mentor) and using such resources as model answers (provided in this book, or on the book's web pages) or a newsgroup such as `alt.comp.lang.learn.c-c++`. But be careful because you will find a great mixture of good and bad in such newsgroups.

Many modern books seem to be written on the basis that the reader needs instant gratification and will only read the text once. That means that we get solid doorstops in which the new information per page is very low. Authors try to find a dozen ways of saying the same thing because they expect the reader to only read a page once. I do not. I expect you to study and that means re-reading as often as is necessary to reach an understanding of what is going on. Take time over it. It took me nine months to write and it took Roberta nine months to understand it. Maybe because of the improvements Roberta has helped me make, it will only take you six months but do not expect to master the contents in much less time. However you will be programming long before you finish this book.

I expect you to work at each chapter and return to earlier chapters as your understanding deepens. That is one of the great strengths of a book as opposed to a training course. The second advantage that a book has is that you can set your own pace. Some things you will grasp quickly, others will take you more time. The things that you understand quickly may well be things that someone else struggles with.

I expect you to work through most of this book with your computer in front of you. Just reading will not be enough; you will need to do. I will assume that you type in the code that I am writing about even when I do not nag you into it. One thing Roberta says fairly early on is that, with hindsight, she made a mistake by skipping some of the code when she was working through my text. Believe her, I rarely if ever waste a student's time with make-work exercises or code that has no value.

Someone with some knowledge of programming casually browsing this book could well be very surprised by some of the material they see in the early chapters. They are used to long, tedious and repetitious tomes that proceed at a snail's pace.

I hope that what you will find in this book is something different. This book aims to explain programming and challenge you to write programs with a limited set of C++ tools. As you progress you will acquire more tools, but the challenge to you to program will be a constant theme.

## What You Will Achieve

Everything that you achieve will be built from simple parts (Standard C++ together with my library). I think you will be surprised how much can be done with simple resources. I certainly find myself playing with Playpen and hardly a day goes by when I do not think of something else I can do with it. That is a key point; what you do with your programming is only constrained by your imagination. For example, it is not that hard to program a computer to play chess, just very hard to write a program to play sensibly, which is why we leave it to experts to write chess-playing programs.

You will also learn techniques to produce simple animation, elementary data processing and numerical work so that your programming basics will be fully rounded out. The main theme of this book is to achieve competence with simple programming and learn that, in essence, it is much simpler than some experts like to make out. Yes, there are arcane corners, weird traps and bizarre features but you do not need to go near them to achieve something that you can be proud of.

Programming should be a rewarding experience. If you do not find it so then either it is not your thing or you have been badly taught. I hope that by studying this book you will discover that you can do it and that you like doing it.

## Notes for Students

Never give up, but learn to ask for help. Have the wisdom to understand the difference between getting help and being lazy. If you do not understand a problem ask for more information but only just enough to point you in the right direction.

If you ever get someone else to write a program for you because you are going to miss a deadline, at least be honest enough with yourself to work at understanding the other person's work. You will already have lost a good deal by not doing the work yourself; do not compound that by not understanding what has been done for you.

## Notes for Instructors

This book is based on many years of classroom experience coupled with over thirty years of programming. Keep focused on what your students require. Impress your students with your qualities as a teacher. That includes the willingness to listen to your students and a desire to understand what they are asking. They do not expect you to know all the answers but they do have a right to expect you to be honest. If you do not know an answer to one of their questions, say so and then take the time to find it.

Please do not destroy the spirit of this book if you use it as a course text. This book is designed to introduce programming basics as a voyage of discovery. The reader is invited to explore what they can achieve with the tools they have been shown rather than constantly hunting for more tools.

Understand that it takes skill and insight to do things in a simple way and appreciate the complicated solutions your students will first offer. But encourage them to look for simpler solutions. It is not enough that a program runs and produces correct solutions; it should also be a clear expression of the solution to a problem.

Do not burden your students with unnecessary requirements. Things like comments should be used constructively and not as some requirement by which you judge the quality of a student's work. If you think

a student is under-commenting their work, wait a couple of weeks and then ask them to explain the program. If they can do so, the comments are probably adequate to their needs. The best documentation of code is the code itself. The more it needs the support of comments the more you should doubt its quality.

## Personal Introductions

### From the lead author

We are going to be spending many hours together so I should introduce myself. You do not have to read this but you might enjoy satisfying your curiosity.

I was born in 1942 (3<sup>rd</sup> June for those who like such trivia) as a first child of six. In 1949 my father went to work in the Sudan and sent his children to the local mission schools in the belief (correct in my opinion) that living in a foreign culture was worth much more than any English primary school education. The school I attended had 2000 pupils, used three teaching languages and there were never more than six pupils there whose first language was English. I had to learn Arabic and Italian. By the time I was nine I was preparing to return to an English prep school so I was also learning Latin from a private tutor and French from my mother who was a fluent French speaker (having been educated in a French convent school).

I started at an English prep school in September 1953 where I added classical Greek to my language studies. I went to Downside – a leading Catholic public school – in April 1956. In October 1960 I went to Merton College, Oxford. There I read a degree in Mathematics, and obtained a third class honours degree. As my tutor said to me several years later, the degree was disappointing but what mattered was that I had got a lot of other things from Oxford such as representing the University at Judo in the annual match against Cambridge in 1961, 1962 and 1963. In 1962 I was president of the Oxford University Judo Club.

I went on to teach mathematics in the early years, becoming Head of Mathematics at Cherwell School, Oxford before taking responsibility for computing in the school. By then I had taught myself to program and had produced several programs for use by my students. In 1982 I implemented the Forth programming language to provide portable programming resources for my students so that they could write programs that ran on their own Sinclair ZX Spectrums as well as the school's Research Machines 380Z. A colleague of mine designed and built hardware to link a Spectrum to a 380Z and I designed a protocol to allow the machines to exchange information over that link.

In 1988 I retired from teaching because the stress of supporting my colleagues with their computing needs had damaged my health. In that same year I joined the C Users Group UK, which later became ACCU. I was Chair of that organization for most of the 1990s as well as editor of its principal publication from August 1990 to December 2001.

In 1990 I became involved in the BSI's panels for standardising C and C++. From there I went on to represent the UK at the ISO/IEC SC22/WG14 (C) and WG21 (C++) committees. During the last few years I have been head of the UK delegation to those workgroups.

If you are interested, I am also an RYA Senior day boat instructor and I play competition Contract Bridge. I have two children as well as a beautiful disabled granddaughter (born in May 2001).

All I know about you is that you want to try out in my world of programming. Welcome, I hope it enriches your life because it has done much for mine, not least allowing me to meet many intelligent and entertaining people. Without those people this book would never have come to be written.

### From the student author

I was born, bred and educated in Oxford and I'm still here. I left school at 16, married young, and had a son and a daughter. During the early years I juggled a wide assortment of part time jobs to fit in with my family commitments; these included telephone operator, barmaid, playgroup leader and working as a butcher's shop assistant in Oxford's wonderful covered market.

When the children were both settled in school I decided to further my education and went to Westminster College as a mature student. Initially I intended to take a teaching degree but I changed to theology. I was thrilled to get a first class honours degree.

After this I joined my husband in his plastic injection moulding factory as a company director and I was responsible for the administration, sales, quality department and general management. This is where I first met Francis who was our company's computer consultant.

When we had built a competent management team I felt that I could return to my studies and I went to Manchester College and took a diploma in theology mainly to learn the Greek and Hebrew necessary for more in-depth biblical studies. I spent the next couple of years specialising in John's Gospel and wrote a book that I have not attempted to get published.

Over the years Francis has continued to help me with my many computer problems. I am not exactly a technophobe but I am rather in awe of the dreaded machines. However, I love the Internet and have found it invaluable for both research and fun. I hope to have my own website eventually and I am slowly designing a website to publish my book and other pieces of writing.

As a development of my interest in all things spiritual I have recently become interested in more esoteric subjects including astrology and kabbalah.

My hobbies over the years have been as eclectic as my work experience and have included ballroom and Latin dancing, archery, gardening, badminton and more recently Tai chi and belly dancing (because now I am 50 I intend to grow old disgracefully) and last but not least having fun with our three grandchildren.

When Francis asked me to be a C++ student I thought he was rather insane. I am scared of computers and useless at math. However, I am sure that if I can learn to program with Francis' help then anyone can, so perhaps I am a good choice after all.

# Acknowledgments

A book is the product of many people in addition to the named authors. An attempt to give an exhaustive list only leads to a feeling of having been slighted by those who have been left out. However there are always a number of individuals who have contributed above and beyond the calls of duty and friendship. In that context I want to publicly acknowledge and thank the following:

Al Stevens ([al@alstevens.com](mailto:al@alstevens.com)) for writing the Quincy IDE and modifying it to better meet my needs even though this book might be considered to compete for some of the potential readership of his book (Teach Yourself C++, 7<sup>th</sup> edition, 0 7645 2644 8).

Garry Lancaster ([glancaster@codemill.net](mailto:glancaster@codemill.net)) for all the many hours he spent implementing Playpen without ever grumbling that it would have been much easier had I given him a complete spec to start with instead of coming up with new items as the work progressed. Garry is among the best MS Windows programmers I know.

Anthony Williams ([anthony\\_w@onetel.net.uk](mailto:anthony_w@onetel.net.uk)) who carefully tweaked the installation code for the CD so that it would make the reader's life as easy as possible.

I also thank all those on the editorial side, most particularly Gaynor Redvers-Mutton who tolerated my fiercely individualistic approach to writing a book.

Finally I should acknowledge the tolerance of my wife, Dulcie, and Roberta's husband, David, without which this book would never have been finished.

